A VHDL IMPLEMENTATION OF THE ADVANCED ENCRYPTION STANDARD-RIJNDAEL ALGORITHM

[1]ASHWINI K S, [2]SHANKARA C

[1]Lecturer, [2] Lecturer

[1]Department of Electronics and Communication Engineering, [2]Department of Electronics and Communication Engineering

[1]Government Polytechnic Chamarajanagar, India, [2]Government Polytechnic Nagamangala, Mandya, India

Abstract: The National Institute of Standards and Technology (NIST) has initiated a process to develop a Federal information Processing Standard (FIPS) for the Advanced Encryption Standard (AES), specifying an Advanced Encryption Algorithm to replace the Data Encryption standard (DES) the Expired in 1998. NIST has solicited candidate algorithms for inclusion in AES, resulting in fifteen official candidate algorithms of which Rijndael was chosen as the Advanced Encryption Standard. The Advanced Encryption Standard can be programmed in software or built with pure hardware. However, Field Programmable Gate Arrays (FPGAs) offer a quicker, more customizable solution. This research investigates the AES algorithm with regard to FPGA and the Very High-Speed Integrated Circuit Hardware Description language (VHDL). Altera Max+plus II software is used for simulation and optimization of the synthesizable VHDL code. All the transformations of both Encryptions and Decryption are simulated using an iterative design approach in order to minimize the hardware consumption. Altera ACEX1K Family devices are utilized for hardware evaluation.

***Index Terms* – Encryption, Decryption, FPGA, DES, AES.**

I. INTRODUCTION

The National Institute of Standards and Technology, (NIST), solicited proposals for the Advanced Encryption Standard, (AES). The AES is a Federal Information Processing Standard, (FIPS), which is a cryptographic algorithm that is used to protect electronic data. The AES algorithm is a symmetric block cipher that can encrypt, (encipher), and decrypt, (decipher), information. Encryption converts data to an unintelligible form called cipher-text. Decryption of the cipher-text converts the data back into its original form, which is called plaintext. The AES algorithm is capable of using cryptographic keys of 128, 192, and 256 bits to encrypt and decrypt data in blocks of 128 bits [1].

Many algorithms were originally presented by researchers from twelve different nations. Fifteen, (15), algorithms were selected from the first set of submittals. After a study and selection process five, (5), were chosen as finalists. The five algorithms selected were MARS, RC6, RIJNDAEL, SERPENT and TWOFISH. The conclusion was that the five Competitors showed similar characteristics. On October 2nd 2000, NIST announced that the Rijndael Algorithm was the winner of the contest. The Rijndael Algorithm was chosen since it had the best overall scores in security, performance, efficiency, implementation ability and flexibility, [NIS00b].

The Advanced Encryption Standard (AES), also known as the Rijndael algorithm, is a symmetric encryption algorithm widely used to secure sensitive data in various applications, including secure communications, data storage, and digital content protection. It was selected as the official encryption standard by the U.S. National Institute of Standards and Technology (NIST) in 2001, replacing the older Data Encryption Standard (DES) [2].

AES is based on the Rijndael block cipher, designed by two Belgian cryptographers, Joan Daemen and Vincent Rijmen. The strength and popularity of AES lie in its ability to provide a high level of security, efficiency, and flexibility. AES operates on fixed-size blocks of data, typically 128 bits, and supports three key lengths: 128, 192, and 256 bits. The encryption process involves multiple rounds of transformations, including substitution, permutation, and mixing operations. The key expansion algorithm generates round keys based on the original encryption key, and these keys are used in each round to transform the data. The number of rounds depends on the key length: 10 rounds for 128-bit keys, 12 rounds for 192-bit keys, and 14 rounds for 256-bit keys.

Due to its simplicity, robustness, and widespread adoption, AES has become the de facto standard for data encryption and is used in a plethora of applications and protocols worldwide, providing a strong foundation for data privacy and security in the digital age. It has undergone extensive analysis by cryptographers, with no known practical attacks against the full AES algorithm, making it one of the most trusted and secure encryption algorithms available today. Its continuous use and prominence in various industries demonstrate the enduring significance and impact of the AES-Rijndael algorithm on modern data security [3].

II. RELATED WORK

AES Proposal: Rijndael. NIST AES Proposal by J. Daemen and V. Rijmen. In this literature, the authors have proposed the cipher Rijndael. The authors first presented the mathematical aspects-based study in order to understand the specifications such as description and design rationale. Later, Cipher implementation aspects and then its inverse are considered. This helps in future works of all the choices of design and resistance treatment for all types of known attacks. The authors have hinted for goals and security claims, Cipher advantages and disadvantages, in which way it has been used for functionality (other than decryption / encryption blocks) [4].

"Advanced Encryption Standard (AES)", National Institute of Standard and Technology (NIST of U.S.), Federal Information Processing Standards Publications (FIPS) are issued by the National Institute of Standards and Technology (NIST) after approval by the Secretary of Commerce pursuant to Section 5131 of the Information Technology Management Reform Act of 1996 (Public Law 104 – 106) and the Computer Security Act of 1987 (Public Law 100 – 235) [5].

"Triple Data Encryption Algorithm Modes of Operation," The publication specifies the tests required to validate Implementations under Test (IUTs) for conformance to the Triple DES algorithm (TDEA) as specified in ANSI X9.52, Triple Data Encryption Algorithm Modes of Operation. When applied to IUTs that implement the TDEA, the TDEA modes of Operation Validation System (TMOVS) provide testing to determine the correctness of the algorithm implementation. This involves both testing the specific components of the algorithm, as well as, exercising the entire algorithm implementation. In addition to determining conformance, the TMOVS is structured to detect implementation flaws including pointer problems, insufficient allocation of space, improper error handling, and incorrect behavior of the TDEA implementation [6].

A DES key has 64 binary digits such as 0's and 1's in which the 56-bit digits are randomly produced and are used in the algorithm. The remaining 8-bit digits are used to detect error not in the algorithm and are set in order to build the each (8 bit) parity of odd key, i.e., there exists an odd number (1's) in each (8 bit) byte [7].

III. PROBLEM STATEMENT

To design and develop a hardware-efficient and high-performance AES encryption and decryption module. The VHDL implementation should be capable of processing 128-bit data blocks using key sizes of 128, 192, or 256 bits, adhering to the AES standard. The main objectives include achieving secure data encryption and decryption with low latency and minimal resource utilization while ensuring robustness against common cryptographic attacks. The design should be scalable, easily configurable for different key sizes, and capable of meeting the throughput requirements for real-world applications such as secure communication and data storage systems.

IV. PROPOSED METHODOLOGY

The first step in the methodology is to understand the AES-Rijndael algorithm thoroughly and break it down into its individual components, such as SubBytes, ShiftRows, MixColumns, and AddRoundKey. Each component will be designed as a separate VHDL module, with appropriate inputs, outputs, and internal logic, implementing the corresponding AES transformation. Next, the main AES encryption and decryption modules will be developed by integrating the individual components in the correct order, following the AES encryption and decryption processes. The main modules will take the plaintext or ciphertext as input, along with the encryption or decryption keys, and produce the encrypted or decrypted output, respectively. Once the VHDL design is complete, it will be simulated using VHDL simulation tools to verify the correctness and functionality of the hardware implementation. Test vectors and known plaintext-ciphertext pairs will be used to validate the design and ensure that the hardware implementation matches the expected results. After successful simulation, the VHDL design will be synthesized to generate a hardware circuit description compatible with the target FPGA or ASIC (Application-Specific Integrated Circuit) device. The synthesis process converts the high-level VHDL code into lower-level gates and registers, optimizing the design for area, speed, and power. Finally, the synthesized hardware description will be implemented on the target FPGA or ASIC device. The performance and security of the AES hardware implementation will be evaluated in this phase, considering factors like throughput, resource utilization, and security against known attacks.

4.1 Encryption Process

Encryption is a process of converting plaintext or readable data into an unintelligible form, called ciphertext, to secure it from unauthorized access or interception during transmission or storage. The purpose of encryption is to protect sensitive information, such as personal data, financial details, passwords, or confidential communications, from being understood by anyone except those with the appropriate decryption key [8].

In an encryption process, an algorithm and a cryptographic key are used to convert the plaintext into ciphertext. The algorithm is a set of mathematical operations that determine how the data is transformed, and the key acts as a parameter for the algorithm, influencing the encryption process. To decrypt the ciphertext back into plaintext and make it readable again, the recipient or authorized party needs the corresponding decryption key.

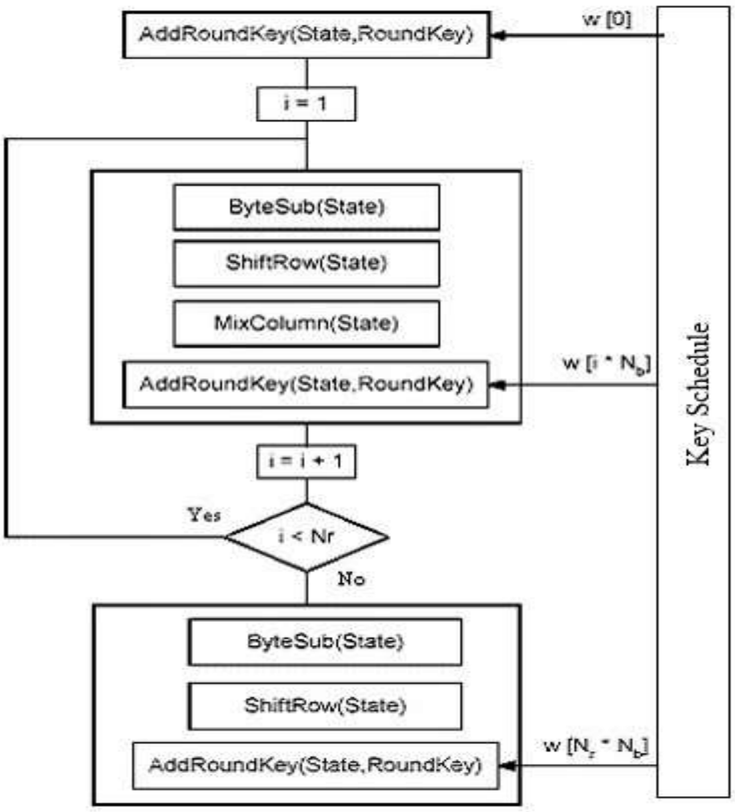

Figure 1. Encryption Process

This block diagram is generic for AES specifications. It consists of a number of different transformations applied consecutively over the data block bits, in a fixed number of iterations, called rounds. The number of rounds depends on the length of the key used for the encryption process.

4.2 Decryption

Decryption is the process of converting encrypted or ciphertext data back into its original plaintext form, making it readable and understandable again. It is the inverse operation of encryption and requires the use of a decryption key, which corresponds to the encryption key used during the encryption process[9,10].

In an encryption process, a cryptographic algorithm and a key are used to transform plaintext into ciphertext. During decryption, the same algorithm is applied, but with the decryption key, to reverse the encryption and recover the original plaintext. The decryption key is a crucial piece of information and must be kept secure and only accessible to authorized parties.

Decryption is a vital aspect of secure communication and data protection. When data is encrypted before transmission or storage, even if intercepted or accessed by unauthorized individuals, it remains incomprehensible without the corresponding decryption key. This ensures that sensitive information, such as personal data, financial details, and confidential messages, remains confidential and safeguarded from potential attackers and eavesdroppers.

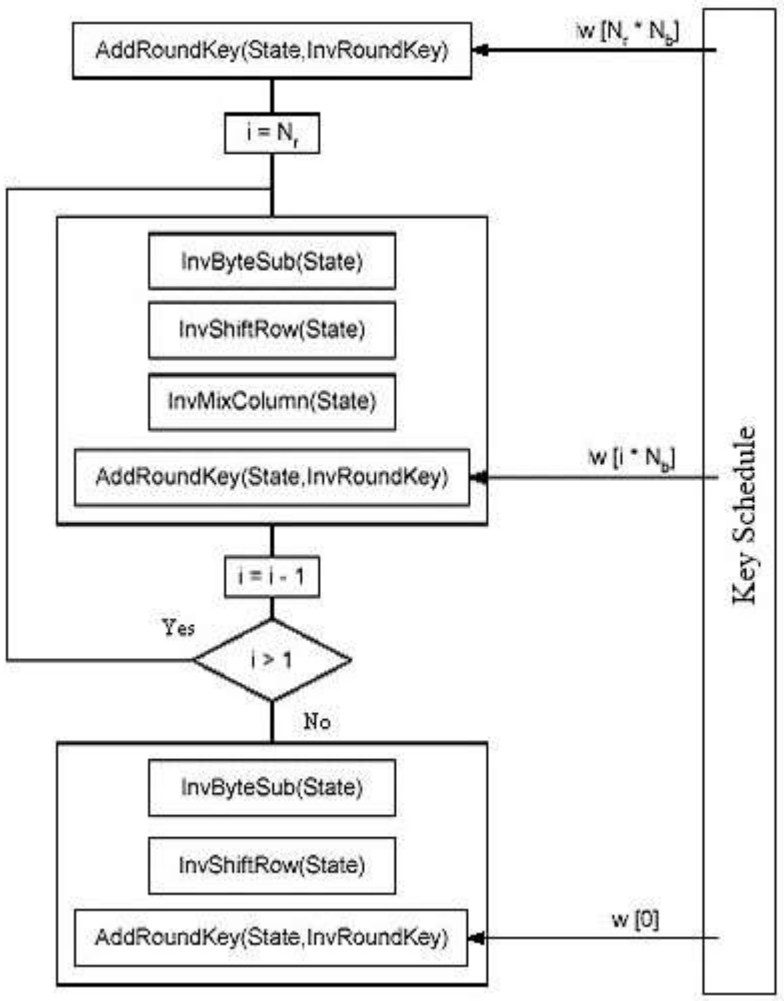

Figure 2. Encryption Process

This process is direct inverse of the Encryption process (chapter2). All the transformations applied in Encryption process are inversely applied to this process. Hence the last round values of both the data and key are first round inputs for the Decryption process and follows in decreasing order.

V. RESULTS AND DISCUSSION

VHDL is used as the hardware description language because of the flexibility to exchange among environments. The code is pure VHDL that could easily be implemented on other devices, without changing the design. The software used for this work is Altera Max+plus II 10.2. This is used for writing, debugging and optimizing efforts, and also for fitting, simulating and checking the performance results using the simulation tools available on MaxPlus II design software.

All the results are based on simulations from the Max+plus II and Quartus tools, using Timing Analyzer and Waveform Generator. All the individual transformation of both encryption and decryption are simulated using FPGA ACEX1K family and EP1K100 devices. The characteristics of the devices are presented in figure 3. An iterative method of design is implemented to minimize the hardware utilization and the fitting is done by the Altera's Quartus fitter Technology.

5.1 Encryption Process

General Characteristics of the ACEX Family				
Device	**EP1K10**	**EP1K30**	**EP1K50**	**EP1K100**
Typical Gates	10,000	30,000	50,000	100,000
Maximum System Gates	56,000	119,000	199,000	257,000
Logic Elements	576	1,728	2,880	4,992
Embedded Array Blocks (EABs)	3	6	10	12
Maximum RAM Bits	12,288	24,576	40,960	49,152
Speed Grades	-1, -2, -3	-1, -2, -3	-1, -2, -3	-1, -2, -3
Package (mm)	Maximum User I/O Pins			
100-Pin TQFP	66			
144-Pin TQFP	92	102	102	

208-Pin PQFP	120	147	147	147
256-Pin (BGA)	136	171	186	186
484-Pin (BGA)			249	333

Figure 3. Basic Characteristics of the ACEX1K Family Devices [4]

In order to allow a full parallel process of the state, it is necessary to implement all the transformations over 128 bits. The most expensive one is the Byte substitution, because it is a table lookup operation, implemented as ROM. Each 8 bits requires a 2048-bit ROM. To process 128 bits, it is necessary 32768 bits. The Key Expansion uses a Byte substitution operation over 32 bits also, so another 8192 bits should be allocated.

The following figure 18 shows the waveforms generated by the 8-bit byte substitution transformation. The inputs are clock of 100ns time period, Active High reset, and 8-bit state as a standard logic vector, whose output is 8-bit S-box lookup substitution. This design utilizes 32% of the area of EP1K100FC484-1, around 1631 logic elements are consumed to implement only 8-bit S-box lookup table. Hence, approximately 20,000 logic elements are necessary to implement the complete 128-bit byte substitution transformation. It can be done by the APEX20K family devices.

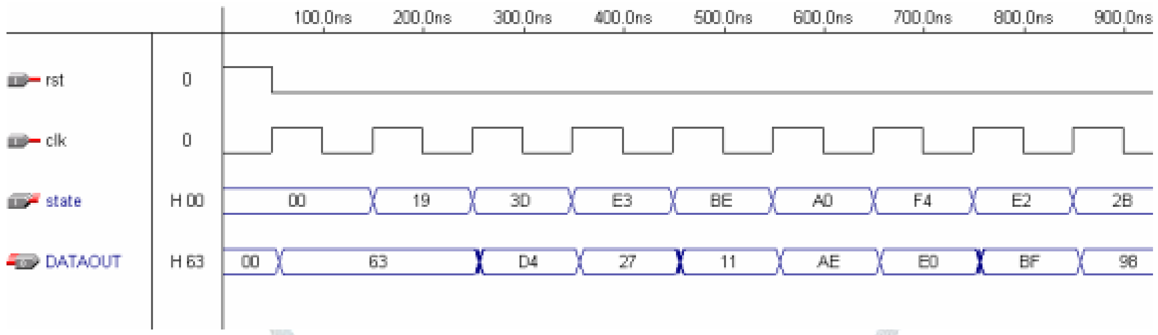

Figure 4. Waveforms of 8-bit Byte Substitution

The following figure 19 represents the waveforms generated by the 8-bit byte substitution transformation. The inputs are clock of 100ns time period, Active High reset, and 128-bit state as a standard logic vector, whose output is shifted as explained in the section 2.3. Design utilizes 2% of the area of EP1K100FC484-1, around 128 logic elements are consumed.

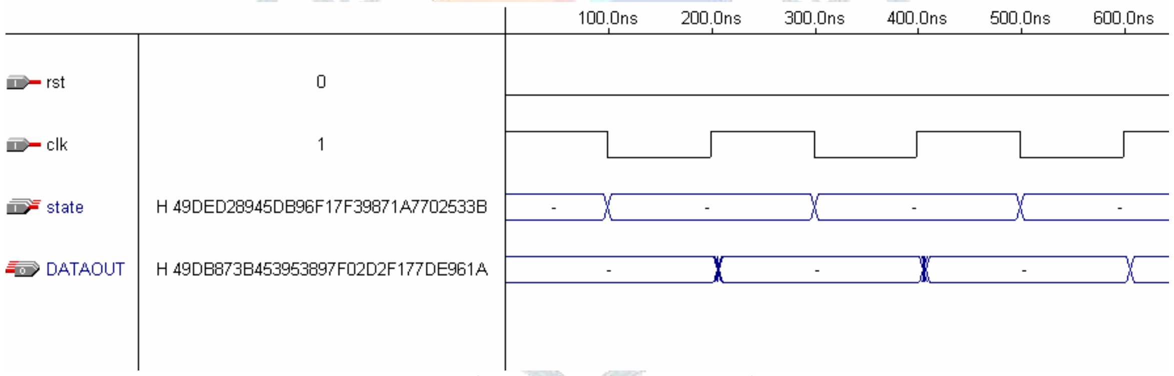
100.0ns
200.0ns
300.0ns
400.0ns
500.0ns
600.0ns
rst 0
clk 1
state H 49DED28945DB96F17F39871A7702533B
DATAOUT H 49DB873B453953897F02D2F177DE961A

Figure 5. Waveforms of Shift Row Transformation

The following figure 20 represents the waveforms generated by the 12 8-bit Mix Columns transformation. The inputs are clock of 100ns time period, Active High reset, and 128-bit state as a standard logic vector, whose output is shifted as explained in the section 2.4. Design utilizes 5% of the area of EP1K100FC484-1, around 156 logic elements are consumed.

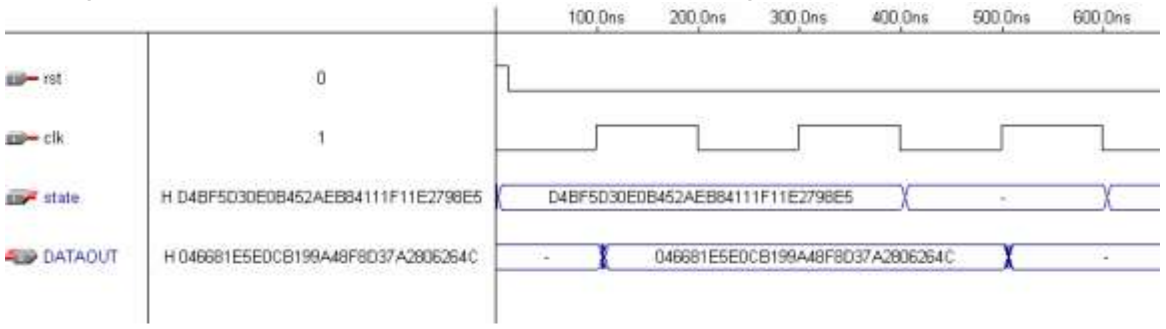
100.0ns
200.0ns
300.0ns
400.0ns
500.0ns
600.0ns
rst 0
clk 1
state H D4BF5D30E0B452AEB84111F11E2798E5
DATAOUT H 046681E5E0CB199A48F8D37A2806264C
D4BF5D30E0B452AEB84111F11E2798E5
046681E5E0CB199A48F8D37A2806264C

Figure 6. Waveforms of Mix Column Transformation

The following figure 21 represents the waveforms generated by the 128-bit Key Schedule Generation. The inputs are clock of 100ns time period, Active High reset, round, and 128-bit state as a standard logic vector, whose output is the 128-bit key for round one is generated. Design utilizes 74% of the area of EP1K100FC484-1, around 3700 logic elements are consumed.

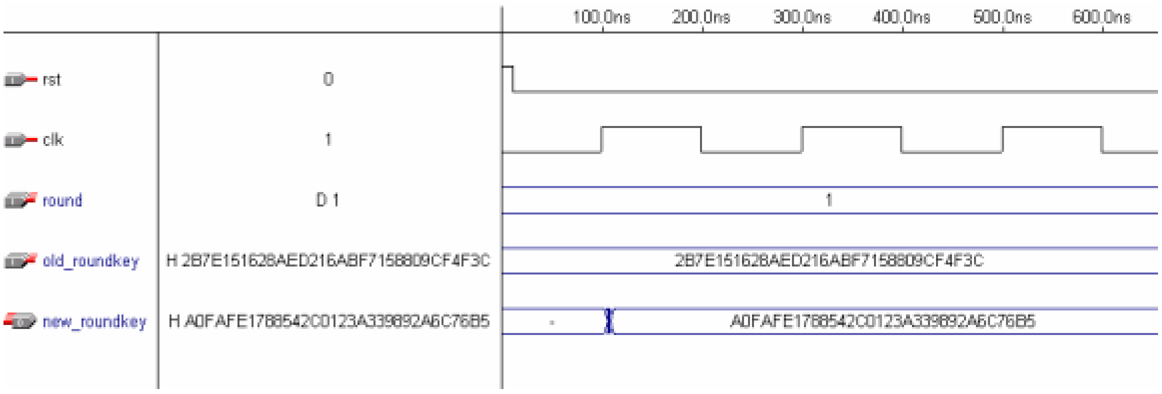
100.0ns
200.0ns
300.0ns
400.0ns
500.0ns
600.0ns
rst
0
clk
1
round
D 1
1
old_roundkey
H 2B7E151628AED216ABF7158809CF4F3C
2B7E151628AED216ABF7158809CF4F3C
new_roundkey
H A0FAFE1788542C0123A339892A6C76B5
A0FAFE1788542C0123A339892A6C76B5

Figure 7. Waveforms of Key Schedule Generation

5.2 Decryption Process

The decryption implementation results are similar to the encryption implementation. The key schedule generation module is modified in the reverse order. In which last round key is treated as the first round and decreasing order follows.

The following figure 22 represents the waveforms generated by the 8-bit byte substitution transformation. The inputs are clock of 100ns time period, Active High reset, and 8-bit state as a standard logic vector, whose output is 8-bit Inverse S-box lookup substitution. This design utilizes 50% of the area of EP1K30TC144-1, around 877 logic elements are consumed to implement only 8-bit S-box lookup table

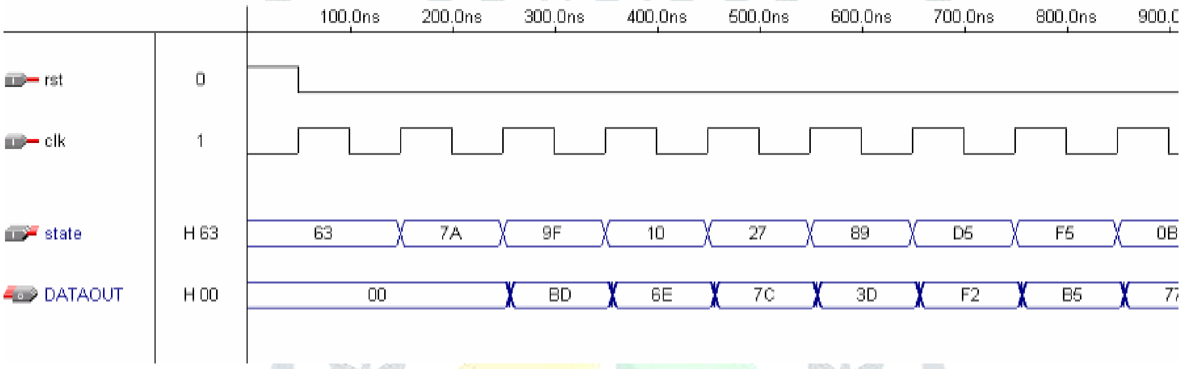

Figure 8. Waveforms of 8-bit Inverse Byte Substitution

The following figure 23 represents the waveforms generated by the 8-bit Inverse byte substitution transformation. The inputs are clock of 100ns time period, Active High reset, and 8-bit state as a standard logic vector whose output is shifted as explained in the section 3.3. Design utilizes 2% of the area of EP1K100FC484-1, around 128 logic elements are consumed.

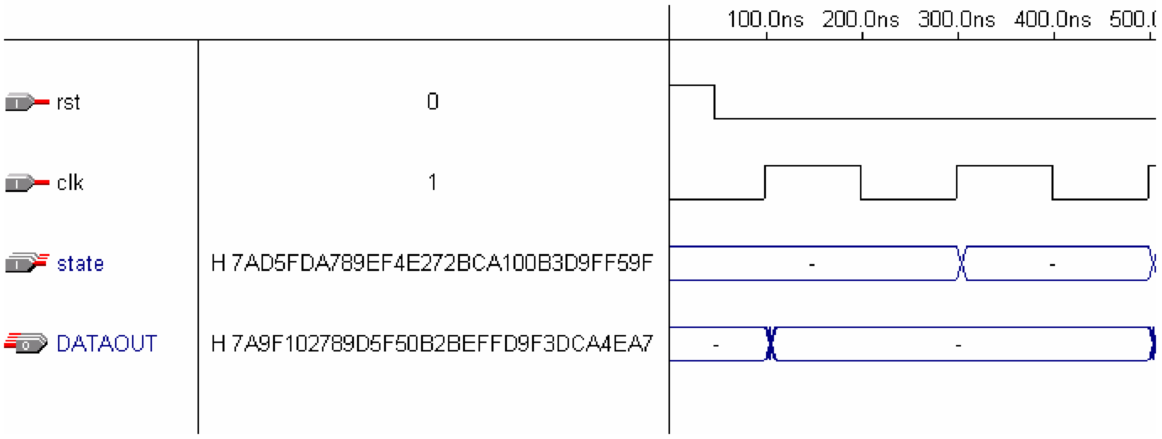
100.0ns
200.0ns
300.0ns
400.0ns
rst
0
clk
1
state
H 7AD5FDA789EF4E272BCA100B3D9FF59F
-
-
DATAOUT
H 7A9F102789D5F50B2BEFFD9F3DCA4EA7
-
-

Figure 9. Waveforms of Inverse Shift Row Transformation

The following figure 24 represents the waveforms generated by the 8-bit byte substitution transformation. The inputs are clock of 100ns time period, Active High reset, and 8-bit state as a standard logic vector, whose output is shifted as explained in the section 3.4. Design utilizes 12% of the area of EP1K100FC484-1, around 624 logic elements are consumed.

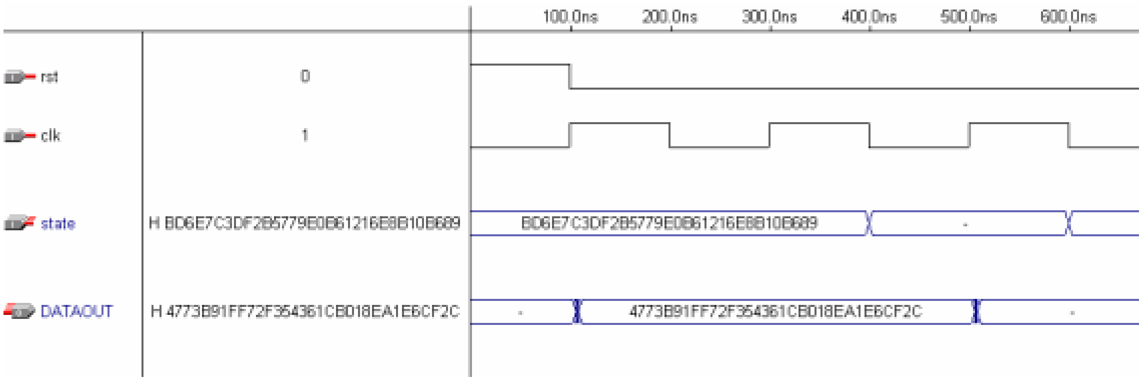

Figure 10. Waveforms of Inverse Mix Column Transformation

VI.CONCLUSION

Optimized and Synthesizable VHDL code is developed for the implementation of both encryption and decryption process. Each program is tested with some of the sample vectors provided by NIST and output results are perfect with minimal delay. Therefore, AES can indeed be implemented with reasonable efficiency on an FPGA, with the encryption and decryption taking an average of 320 and 340 ns respectively (for every 128 bits). The time varies from chip to chip and the calculated delay time can only be regarded as approximate. Adding data pipelines and some parallel combinational logic in the key scheduler and round calculator can further optimize this design. The software can be used to program an FPGA device with the Advanced Encryption Standard (with key widths of 128-bit, 192-bit and 256-bit on a single FPGA device). The Field Programmable Gate Arrays (FPGAs) provide the more customized and quicker AES solution. The research explores the AES algorithm with the help of Verilog as the hardware description language and Field Programmable Gate Arrays (FPGAs). For optimization and simulation purpose, Xilinx ISIM 14.7 software is used. The simulation of both decryption and encryption is done by using an approach of iterative design to minimize the consumption of hardware. For hardware evaluation is carried out by the help of Xilinx Spartan 6 devices.

VI. ACKNOWLEDGMENT

The authors will like to express sincere thanks and appreciation to his esteemed mentor and supervisor for their ongoing inspiration, technical support, guidance, encouragement, and insightful suggestions that enabled him to undertake this research work.

REFERENCES

[1] FIPS 197, “Advanced Encryption Standard (AES)”, November 26, 2001 http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf
[2] Daemen and V. Rijmen, “AES Proposal: Rijndael”, AES Algorithm Submission, September 3, 1999 http://www.esat.kuleuven.ac.be/~rijmen/rijndael/rijndaeldocV2.zip
[3] ALTERA. Max+plus II VHDL. San Jose. Altera, 1996
[4] ALTERA “ACEX1K Embedded Programmable Logic Family Data Sheet”, pdf files, http://www.altera.com/literature/ds/acex.pdf (May 2003)
[5] ALTERA High-Speed Rijndael Encryption/Decryption Processors, http://www.altera.com/literature/wp/wp_hcores_rijnfast.pdf
[6] Marcelo B. de Barcelos Design Case, “Optimized performance and area implementation of Advanced Encryption Standard in Altera Devices, by, http://www.inf.ufrgs.br/~panato/artigos/designcon02.pdf
[7]“FPGA Simulations of Round 2 Advanced Encryption Standards” http://csrc.nist.gov/CryptoToolkit/aes/round2/conf3/presentations/elbirt.pdf.
[8] http://en.wikipedia.org/wiki/Extended_Euclidean_algorithm
[9] Tilborg, Henk C. A. van. “Fundamentals of Cryptology: A Professional Reference and Interactive Tutorial”, New York Kluwer Academic Publishers, 2002
[10] Peter J. Ashenden, “The Designer's Guide to VHDL”, 2nd Edition, San Francisco, CA, Morgan Kaufmann, 2002